PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/60

A1

(11) International Publication Number: WO 00/70511

(43) International Publication Date: 23 November 2000 (23.11.00)

(21) International Application Number: PCT/AU00/00468

(22) International Filing Date: 18 May 2000 (18.05.00)

(30) Priority Data:
PQ 0422 18 May 1999 (18.05.99) AU

(71) Applicant *(for all designated States except US)*: KCS AUSTRALIA PTY. LTD. [AU/AU]; Level 4, 51–55 Mountain Street, Ultimo, NSW 2007 (AU).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: WOLFE, Jonathon [AU/AU]; Level 4, 51–55 Mountain Street, Ultimo, NSW 2007 (AU). WHITE, Teresa [AU/AU]; Level 4, 51–55 Mountain Street, Ultimo, NSW 2007 (AU).

(74) Agent: F. B. RICE & CO.; 605 Darling Street, Balmain, NSW 2041 (AU).

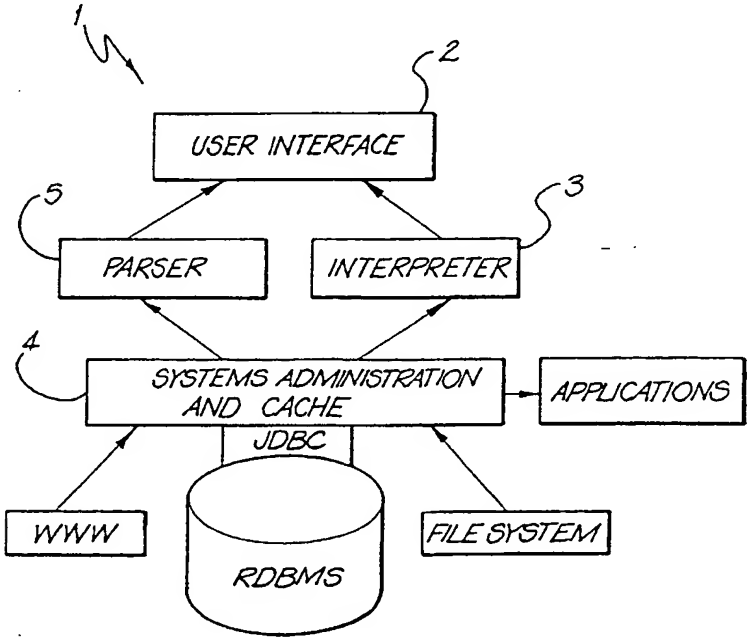
(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: A WEBSITE DEVELOPMENT SYSTEM

(57) Abstract

This invention concerns a website. In another aspect it concerns a development system for the creation, maintenance and publishing of websites on the World Wide Web (WWW). Websites typically contain a number of pages through which a visitor to the site is able to surf using navigation functions. The pages themselves may contain text, graphics, photographs, sound and video as well as the navigation locations. Any selected parts or combinations of parts are defined as content objects which are arranged in a hierarchy with links to other content objects using URLs which are derived by the location of the content object in the hierarchy. Templates define individual elements in a web application without regard for the specific web pages on which the elements might appear, and are also defined as content objects. The URL of each template instance includes a pointer to one or more parent templates. The templates are constructed using taglets which define the content objects that are to be used to build a page. All the templates and template instances are stored in a database and when a page is requested, the appropriate template is activated and the taglet HTML is interpreted to request the template instances required to build the page. The development system includes an interface to communicate with client browsers to receive webpage content and webpage templates, and to deliver websites to a user. A parser to convert webpage content received from the interface into predefined parts which are defined as content objects which are all instances of templates, the parser also operates to convert templates, which are constructed using taglets which define the content objects that are to be used to build a page, into content objects. And an interpreter to process content objects, when a webpage is requested by activating the appropriate template and interpreting the template taglets to request the template instances required to build the page.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

A WEBSITE DEVELOPMENT SYSTEM

Technical Field

This invention concerns a website. In another aspect it concerns a development system for the creation. maintenance and publishing of websites on the World Wide Web (WWW). Websites typically contain a number of pages through which a visitor to the site is able to surf using navigation functions. The pages themselves may contain text, graphics, photographs, sound and video as well as the navigation locations.

Background Art

Static websites use file systems as the data repository. The pages are held on the webserver as individual HTML pages. This model allows many different people to add content to the website quite simply by providing new HTML pages to the webserver. The files to be served can be managed easily and it is very easy to set up a website. This ease of set up and management contributed significantly to the growth of the WWW.

As a static site grows larger complexity grows exponentially as more files are added. A site may have two hundred files with some internal links and is manageable with simple tools. However some of the largest static sites have ten thousand files or more. with each file having many links to other files within the site. The relationships among the files become so numerous and complex that no one person can track them and no tool can solve the problem. To customise a large website to different visitors requirements where the content is in static files involves coding complex and expensive CGI scripts.

Dynamic architectures use databases to hold content and use programming or scripting languages to define classes of dynamically generated pages. In a template based dynamic architecture a template, which is a program, extracts the same collection of data from the database and displays the same page to every user who surfs to it. Dynamic page generation solves most of the file-based problems because it defers all decisions about content relationships to the moment when the user's browser requests the page.

Dynamic pages address the need to constantly change content by allowing content to be updated in the database. However, the only way to

give every user a unique view of every page involves complex programming to generate a customized page for each user. The production of the code to produce the desired combination of HTML and the information from the database in a dynamic webpage is very tedious and clumsy. This work needs to be done by a specialist programmer who is unlikely to have the necessary artistic skills so prolongs the process. Such systems require large amounts of CPU power to execute and cannot scale to provide an acceptable level of performance on affordable hardware.

Dynamic architectures built on programming languages, such as C, C++, or Java, are even worse. In these systems developing web applications is tedious and time consuming, because everything becomes a programming exercise. Their fundamental problem is that they operate at levels of abstraction that are much too low. For example, just generating simple HTML output requires programming a series of "print" statements to generate the HTML tags. This makes it impossible for designers to handle layout considerations without knowing about programming, and precludes the use of HTML layout and design tools to assist with template development and maintenance.

These difficulties have, in part, led to the definition of Taglets. Taglets are a generic way of defining special HTML tags that the server understands and processes rather than sending directly to the browser. Taglets allow the server programmer to define custom, site/purpose specific server-side functionality (in Java) which may be used by HTML writers who have no knowledge of computer programming but are familiar with the syntax of HTML. Taglets are in common use in products such as JRun. Traditional taglet implementations rely on translating the taglet HTML into java source code and then attempting to "compile" the java source code automagically, all at the time the page is requested.

Summary of the Invention

The invention is a website, including:

One or more pages each of which comprises one or more logical parts that are visible to a user. Each of the parts may be any page content, such as a piece of information, text, images, a script or any other type of digital media.

Any selected parts or combinations of parts are defined as content objects.

The content objects are arranged in a hierarchy with links to other content objects using URLs which are derived by the location of the content object in the hierarchy.

The boundaries of the content objects are also defined by the URLs.

Templates, which define individual elements in a web application without regard for the specific web pages on which the elements might appear, are also defined as content objects, and the URL of each template instance includes a pointer to one or more parent templates.

The templates are constructed using taglets which define the content objects that are to be used to build a page.

All the templates and template instances are stored in a database and when the page is requested, the appropriate template is activated and the taglet HTML is interpreted to request the template instances required to build the page.

By interpreting the taglets at page request time they become much more useable for average HTML content creators since any errors generated by interpreting a taglet can be presented in a format which can be easily interpreted by relatively unskilled users. This can be contrasted with past practice where taglets have been compiled, and at page request time the user might see java compilation errors when attempting to use the Taglet which only make sense to the taglet author.

In a content-objects approach, templates define individual elements in a web application without regard for the specific web pages on which the elements might appear. Some examples of content objects include index pages, header images, and navigation bars. With a content object approach, template documents define actual pages in the web application by specifying which other content objects will be used to build the page. Individual content objects might be used on many types of pages throughout the web site.

Content objects have the following advantages over known page-template architectures:

They allow an application to be built in terms of individual functional elements. For example, on a typical web page there might be four content objects: a navigation bar global to the application, a navigation bar specific to

the current context, a "body" section of the page containing the current content, and a "teaser" section containing links to other related content. This concept applies to all web application types, whether pure publishing, customer support, catalog or commerce.

Content objects facilitate reuse and dynamic assembly, by making it easy to "factor" an application into simple, reusable, pieces. This reduces application development time.

Content objects provide well-defined interfaces to each other, allowing them to be combined in new ways without fear of interference from hidden assumptions. This is fundamentally different from a static "include" model, which is sometimes found in template architectures.

Content objects can be updated independently, allowing pieces of the application to evolve at different rates.

Content objects are ideally suited to assisting user navigation.

Take the example of a dealer application, which wants to display reports to dealers in an extranet environment. Each report may need to be formatted similarly - this would be a classic example of what template-based approaches are best at. However, each customer could also receive a completely dynamic element within the reports, to provide the customer with personalised information that is specifically tailored for his or her preferences and context. Maybe some customers want to use a Java applet to assist their navigation, while others want a simple Next/Previous button.

These sorts of variations break the simple dynamic template model, forcing application developers to perform large programming tasks to model the desired behaviour. With a component model, there would be three components in this example: one for the reports, one for the Java applet navigation system, and one for the Next/Previous navigation system. Dynamic assembly support would allow a content object-based system to combine the appropriate content objects together and provide each user with a customized view of the page.

By breaking down the document into content objects we are removing the requirement of many document management systems to classify the information at the time of inclusion into the data repository. The document information can be stored at a very high level of granularity and these objects reused many times. Much of the difficulty that modern organisations have is reinterpreting the information that they already have in their possession

because when that information was created it was done so in one context without the realisation that it could be equally as useful in another context some time in the future. By breaking documents down into content objects we are able to release them from their context and just analyse and use the information that they contain.

It is also possible to gain considerable synergy by combining different objects in different ways. If for example each heading of a document could be stored as a different content object within the data repository. Then each of the heading objects could be listed against a certain set of criteria along with the author to quickly see the representation of this subject area in the data repository and also who had contributed this information. So the combination of a heading object and an author object provides new and important information that is not contained within the original stand alone document at the time of creation.

Another example of this might be a contract that is negotiated between two corporate entities, the myriad of documents are produced and fed through a definitive process with an obvious outcome. Some time later these contracts are subject to litigation on some point of implementation it will then be the job of the legal team to examine those documents in a different context. This can easily be done where a content object approach has been taken at the time of creation. An ad hoc query against the content objects in the data repository could be constructed to extract the information automatically, and other objects may be added to the query statement to deliver the correct context.

Another situation where this approach would be utilised is in developing site structure and its components to build a time based view of the site. A tool that allowed the user to see the site as at a certain date would allow a legal team to see what information was available to a decision maker when they made the decision. This sort of functionality would obviously be very useful in reconstructing a chain of events.

Another advantage of the content object approach with a single repository of components is that it will efficiently serve many more people because of the ability to combine the content objects in many different ways given sufficient granularity.

With sufficient granularity it is possible to build a data repository in a cost effective and efficient manner that will service the great majority of

users of the information resource. By examination of the website from the point of view of having several different constituencies then it is possible to devise a communication message along identified bands of interest.

For example a Finance Director visiting a site about office machines will want to see information about the product as well as information relating to the total cost of ownership of the product. Content authors can make assumptions about the type of content that the Finance Director will want to see and pull together these content objects in defined ways. If however the Finance Director wanted to stray into other content areas that he is particularly interested in then the site can be constructed uniquely along those lines. The unique combination of different content objects is easily achieved at the time of request against the data repository.

In another aspect the invention is a website development system for the creation, maintenance and publishing of websites, including:

An interface to communicate with client browsers to receive webpage content and webpage templates, and to deliver websites to a user. The webpage content may be in any form, such as a piece of information, text, images, a script or any other type of digital media.

A parser to convert webpage content received from the interface into predefined parts which are defined as content objects which are all instances of templates. Also, to convert templates, which are constructed using taglets which define the content objects that are to be used to build a page, into content objects. Where the content objects are arranged in a hierarchy with links to other content objects using URLs which are derived by the location of the content object in the hierarch. The boundaries of the content objects are also defined by the URLs, and the URL of each template instance includes a pointer to one or more parent templates.

An interpreter to process content objects, when a webpage is requested by activating the appropriate template and interpreting the template taglets to request the template instances required to build the page.

There are various roles that people take on when using the system:

A casual surfer who does not distinguish the site from any other website.

A content author who has access to the administration module within the system. They are able to access the directory structure of the site and surf to and edit content that is owned by them. This person would not typically

have any technical skill. They create static content that they would enter into the website via a template instanciation that they have created.

An application programmer who has access to the administration module within the system and is able to access the directory structure of the site and to edit content that is owned by them. These people may construct templates and dynamic elements within the templates. Typically they will be contributing content extracted from a legacy database or some other application.

A technical template author with some HTML coding experience. This person has access to the administration module within the system and is therefore able to access the directory structure of the site and surf to and edit content that is owned by them. They construct templates, this involves defining the structure and elements and boundaries to the content objects.

A website administrator who is a skilled person who controls the access list and the group user model. They are people who ensure that the server continues to operate efficiently and performs the overall tasks required.

The generation of HTML has varied requirements depending on the skill level of the operator. Highly skilled practitioners edit raw HTML in an environment where their work can be quickly visualised. The lack of editors capable of dealing with frames also force skilled users to operate in an environment where they work with the raw HTML.

HTML offers the inexperienced user the ability to create presentably formatted data containing graphics and other multimedia in a form that can be transported to anyone with access to a web browser. Generally using a WSYWYG HTML editor, the provision of content can be pushed out to those users who "own" the information that is being provided.

It can be seen that the system enables distribution of content creation and centralisation of website management and control over the website publishing process. In addition it permits designing and delivering relevant and personalised web based content to visitors to a site.

The relationship from the template instance to the parent template is maintained only by the URL link. This means that the template has no knowledge of any of it's children. If the either the template or template instances are renamed, the link is not affected because all the objects use

unique identifiers internally, and the URLs (which contain the name) are used externally only.

A secondary template is a template that references existing content and combines it in a new way or in combination with new content. By way of example the detailed specifications of a product that appear on one web page could be reused in the whitepaper that describes the performance of the product. The secondary template references existing content objects by including the URL for the required objects.

The complete content of each webpage is held in the database and can be used on the webpages in a many combinations and positions. The content of the webpages can be developed independently of each other and it is a simple matter to design a webpage using static and dynamic (continually being updated from the database) elements. In this way large teams of programmers are able to produce a complex website working in parallel. The programmers are able to view the entire website and see how their work will impact the website at any level of detail.

The website Production Manager has tools for 'Locking' to prevent conflicting edits and 'Synchronisation' to make the integration of content from various programmers a simple process.

To produce a personalised webpage drawing from a customer database is relatively simple using 'conditional include'.

The system provides the information through any web browser and content can be created in a WYS/WYG format. There is provision to roll back to previous versions simply as well as producing multiple versions of the website, for instance a variation in the language used and which webpages components are provided in response to enquires from various sources. The customer can also add to the database if approved by a method that does not effect any of the other data and thereby the customers input becomes part of the website.

The production of websites is therefore faster, contain more timely changes and as a result be of better quality and more satisfying to the user.

By implementing the concept of content objects combined with a compact and powerful scripting language the invention is able to provide an architectural framework that is very flexible. As content objects can be used and reused in different contexts the author is able to quickly and easily update the existing site structure without reference to any programming

expertise. The document template structure is defined using taglets which are simple HTML extensions, and there are no forms to code or scripts to write. In use the content objects are manipulated through the interpreter and cache and delivered to the site visitor.

The document templates are defined within the HTML by reference to a taglet such as the <A HREF = "max-template component"> tag that in the majority of cases defines content object boundaries. An HTML programmer can insert the required tags into the page structure. They can quickly and easily deploy templates that reference a variety of content objects in different contexts.

Effective revision control and development management software is necessary in the development of large sites that are updated frequently. The problem can become one of the largest headaches in the development of web sites. Many time consuming errors can be made that require large efforts to repair. The development system needs to be able to control access to all parts of the system.

There are two primary requirements to a revision control system.

Firstly, a revision control system must maintain a repository containing information about the files that have existed in a site. It must know which files are required to serve the site and be able to inform the user on which files are active and which are not.

Secondly, the revision control system must maintain the data. It should contain the change history of all the files in the site.

The system supports the development of complex websites by allowing multiple contributors to make up the web development team. The system manages the permissions and responsibilities of the site development team to the components of the website. The site is developed within the context of the website not as a separate entity. The process of content creation can be removed from the responsibility of the site development team so they can direct their energies towards building site structure and navigational design.

With a security model implementation to the component level it is possible to push the responsibility for content creation and updating back to the people responsible for the material while maintaining control of access to that content.

The system can restrict access to various objects within any page in order to efficiently divide the development process, so the programmer can

concentrate on programming, the layout HTML programmer can focus on their expertise and the graphic artist is restricted to manipulating images and look and feel. The system uses the database to lock down the various elements of the site to the component level enabling the multiple skills and competencies to be applied to the development of the website at the same time. For example a graphic artist can alter an image on the page while a programmer is altering the executable code embedded within the same page.

Using this methodology, a series of templates can be constructed that present to the user only the information required to perform their assigned function within the system. The templates can be constructed in HTML with minimal extensions.

The concept behind distributed publishing is to ensure the widest possible distribution. This is achieved by only utilising standard Internet technologies so all the content can pass through the various security measures without risk or interference. The template structure therefore also allows the combination of both static and dynamic elements into the same page or view.

Brief Description of the Drawings

An example of the invention, known as websiteMAX, will now be described with reference to the accompanying drawings, in which:

Figure 1 is a schematic view of a system embodying the invention.

Figure 2 is a screen shot of websiteMax displaying an empty directory.

Figure 3 is a screen shot of creating a new HTML template called 'product.hmt'.

Figure 4 is a screen shot of the directory now displaying the 'product.htm' template, which contains 280 bytes of default content.

Figure 5 is a screen shot of the HTML source code of the 'product.htm' template, showing its old content.

Figure 6 is a screen shot of the contents of a local file called 'a_catalogue_entry_template.htm' published to websiteMAX, overwriting the default content of the 'product.htm' template.

Figure 7 is a screen shot of the websiteMAX directory displaying 'product.htm' with its new content, which has expanded the file size to 618 bytes.

Figure 8 is a screen shot of the HTML source code of the 'product.htm' template. showing the new content.

Figure 9 is a screen shot of 'product.htm' viewed using the websiteMAX 'Browse' function.

Figure 10 is a screen shot of creating a new instance of the 'product.htm' template called 'car.htm'.

Figure 11 is a screen shot of websiteMAX now displaying both files in its directory - 'car.htm' has 150 bytes of default content.

Figure 12 is a screen shot of the default content of 'car.htm' viewed using the websiteMAX 'Edit' function.

Figure 13 is a screen shot of the default content of the 'car.htm' instance viewed using the websiteMAX 'Browse' function.

Figure 14 is a screen shot of changing the content of the 'car.htm' instance using a WYSIWYG HTML editor (Netscape Composer).

Figure 15 is a screen shot of the updated content of 'car.htm' viewed using the websiteMAX 'Browse' function.

Figure 16 is a screen shot of websiteMAX displaying both files in its directory - 'car.htm' is now 356 bytes in size after being updated.

Best Modes of the Invention

The concept of websiteMAX is to provide a platform to facilitate the development of large static/dynamic sites. The platform is designed to provide a logical extension to the current implementation of websites that contain static and dynamic content.

On a conceptual level. the websiteMAX system architecture 1 is broken into four components as shown in Figure 1: the user interface 2, the interpreter 3, the cache 4 and the parser 5. The user interface 2 is implemented on the client machine, while the remaining elements are delivered by the websiteMAX server.

The main physical components of websiteMAX are the websiteMAX client which is accessed simply using any standard web browser. It allows users to add, delete. copy, move and modify web pages in a website and automatically open a page in any HTML editor. Alternatively, users can create and modify pages using their favourite HTML editor;

The websiteMAX server allows both traditional static serving from content generated automatically from a file system and dynamic serving from content stored in a RDBMS.

User Interface

A fundamental requirement of the user interface for the websiteMAX system is to allow use of the system from a web browser. This has a number of immediate advantages:

(i) it allows ready access to the websiteMAX system using software already deployed on a large number of desktops. This also obviates the need for low end users of the system to load yet another piece of software to be maintained on a client machine.

(ii) enabling client machines to use websiteMAX may be achieved by loading a readily available web browser.

(iii) all interaction between the websiteMAX server and the client occur through a web server communicating via HTTP. As a result, websiteMAX will communicate through corporate firewalls with no reconfiguration requirements or displacement of existing proxy servers cause. This approach is critical in an intranet or extranet environment where there are security systems such as firewalls that prevent access via non-standard communications channels.

(iv) the user interface satisfies the requirements of the novice to HTML and the expert.

For high end users of the websiteMAX system, there is also a JAVA publishing tool which runs external to the browser. The publishing tool provides some interface functionality improvements for the interface, but will primarily provide productivity gains which will be welcomed by experienced users of the system.

Interpreter

The interpreter provides part of the serving functionality of websiteMAX.

The websiteMAX interpreter processes objects which have been defined in the cache and reconstitutes them as HTML. It also processes any of the special websiteMAX tags which have been added to enhance the functionality of the HTML. WebsiteMAX offers additional functionality such

as being able to define variables in a page. loop statements and conditional statements.

The interpreter does not adopt the approach of other systems on the market which are attempting to provide an environment for developers to produce dynamic applications. The command set is slim and only provides simple functionality that is coded by experienced HTML programmers.

Applications. on the other hand, are written in the appropriate language and environment for software development and then integrated with the websiteMAX system through an appropriate directive in the HTML code. The interpreter performs the function of identifying the directives in the HTML and executing the relevant application to provide the content for that section.

In this manner HTML and application development are clearly delineated in their respective places in the development process.

Cache

The cache plays a pivotal role in the websiteMAX system design.

The cache is essential to provide the performance required so that the system is scalable. The cache's goal is to minimise the number of times that the system needs to retrieve data from the database as it may be a performance bottleneck with the high level of deconstruction that is performed on individual pages.

In addition, the cache maps out the relationships between objects in the database. This allows the system to deliver a complete page quickly - pulling all the data required directly from the server.

The nature of the database structure provides maximum flexibility for future application design but does not necessarily provide optimal performance without the cache.

The implementation of the cache has been carried out in a completely general manner such that the cache can be implemented to operate over a file system in addition to the database.

The cache connects to the RDBMS through multiple connections to provide better performance. There is a layer in between the cache and the database, which defines the concept of a pool of connections. The cache is not concerned about the number of connections. The pool just supplies requests for drivers to connect to the database for an operation. One

connection is reserved for the reading from the database because reading is relatively fast when compared to writing. The other connections are used for those users that wish to update data. There can be any number of connections in the pool and the number of connections utilised is a tunable parameter.

Parser

The parser provides facilities for breaking the HTML into the smaller objects that are stored in the cache and then into the database.

It also takes links and turns them into object identifiers. If there is a file that has a link to another file in websiteMAX, a link identifier comes through the parser, websiteMAX finds the object ID of the referenced document and puts that into the cache.

The inherent portability of Java makes light of cross-platform issues. Development of the websiteMAX server has taken place in three different environments (Win 95, Windows NT and Solaris) with no portability issues being discovered.

Interpreted languages have always had significant advantage over compiled languages in the area of debugging. Source code debugging and profiling is efficient and well implemented in Java development tools allowing for the enhanced productivity that this capability provides.

Java allowed the use of JDBC, which provides a standardised interface to RDBMS's. Individual JDBC drivers are still implemented by vendors, allowing for reasonable performance, but allow for Java code to be written in an RDBMS independent manner. WebsiteMAX has been tested under Oracle, Sybase and many other database systems.

The Publishing Process

The Publishing Process will now be described with reference to Figures 2 to 16. Figure 2 shows an empty directory. In Figure 3 a dialogue box entitled "New Html Template" has been brought up, and the text 'product.htm' has been written in to title the new template.

In Figure 4 the directory now displays the product.htm template which contains 280 bytes of default content. Figure 5 shows the HTML source code for the content. This content is then changed by calling up the "Publish File"

dialogue box, shown in Figure 6, and entering the file name 'a_catalogue_entry_template.htm'. Hitting the OK button publishes this content to websiteMAX and overwrites the default content of the product.htm template. Figure 7 shows the directory in which the content of product.htm has been expanded to 618 bytes, and Figure 8 shows the HTML of those bytes.

Figure 9 shows the result which is viewed using the websiteMAX Browse function to view product.htm.

In Figures 5 and 8 websiteMAX tag attributes, such as

href="max-template: x".

A new template instance can now be created using the published HTML template for product.htm as the parent file for the new template instance. In the dialogue box entitled "new HTML template instance" which is brought up in Figure 10, the name of the parent file 'product.htm' is entered and the name of the new template instance 'car.htm' is entered. Figure 11 shows both files in the directory, and car.htm can be seen to have 150 bytes of default content.

The websiteMAX edit function can then be invoked, as shown in Figure 12 to view the default content of car.htm. Figure 13 shows the default content of car.htm instance when viewed using the websiteMAX browse function.

The content of the car.htm instance can then be changed using any WYSIWYG HTML editor such as Netscape Composer, and details are shown entered by the name, number and description of the car instance. After they have been entered, Figure 15 shows the updated content which is viewed using the websiteMAX browser function. A return to the directory as shown in Figure 16, shows both files with car.htm having 356 bytes after being updated.

When you publish a file from your local system to websiteMAX, it automatically replaces one of the files currently stored in the current websiteMAX directory. The file being published will also take the name of the file it replaces, that is when you publish update.htm as a replacement for current.htm then the published file will be listed in the current websiteMAX directory as current.htm even though its content has been replaced with that of update.htm. The rename option can be used from the file memory to rename the files if necessary.

It can be seen that the publishing process avoids the use of formlets. Low level users are able to edit template instances in WYSIWYG as shown above, but high level users can construct the templates in raw HTML.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

CLAIMS:

1. A website, including:

one or more pages, each of which comprises one or more logical parts that are visible to a user, where each of the parts display media content, and any selected parts or combinations of parts are defined as content objects;

the content objects are arranged in a hierarchy with links to other content objects using URLs which are derived by the location of the content object in the hierarchy;

the boundaries of the content objects are also defined by the URLs;

templates, which define individual elements in a web application without regard for the specific web pages on which the elements might appear, are also defined as content objects, and the URL of each template instance includes a pointer to one or more parent templates;

the templates are constructed using taglets which define the content objects that are to be used to build a page;

all the templates and template instances are stored in a database and when a page is requested, the appropriate template is activated and the taglet HTML is interpreted to request the template instances required to build the page.

2. A website according to claim 1 where the taglets are interpreted at page request time.

3. A website according to claims 1 or 2 where the medial content is a piece of information, text, images, a script or any other type of digital media.

4. A website according to claims 1, 2 or 3 where the templates define individual elements in a web application without regard for the specific web pages on which the elements might appear.

5. A website development system for the creation, maintenance and publishing of websites, including:

an interface to communicate with client browsers to receive webpage content and webpage templates, and to deliver websites to a user;

a parser to convert webpage content received from the interface into predefined parts which are defined as content objects which are all instances of templates, the parser also operates to convert templates, which are constructed using taglets which define the content objects that are to be used to build a page, into content objects;

where the content objects are arranged in a hierarchy with links to other content objects using URLs which are derived by the location of the content object in the hierarch:

the boundaries of the content objects are also defined by the URLs, and the URL of each template instance includes a pointer to one or more parent templates;

an interpreter to process content objects, when a webpage is requested by activating the appropriate template and interpreting the template taglets to request the template instances required to build the page.

6. A website development system according to claim 5, where the webpage content is a piece of information, text, images, a script or any other type of digital media.

7. A website development system according to claim 5, where the system provides information through any web browser, and content is created in a WYS/WYG format.

8. A website development system according to claim 5, where there is provision to roll back to previous versions as well as producing multiple versions of a website.

9. A website development system according to claim 5, where the document template structure is defined using taglets which are simple HTML extensions, and there are no forms to code or scripts to write.

10. A website development system according to claim 5, where in use the content objects are manipulated through the interpreter and cache and delivered to the site visitor.

11. A website development system according to claim 5, where the document templates are defined within the HTML by reference to a taglet that defines content object boundaries.

12. A website development system according to claim 5, where the system manages the permissions and responsibilities of a site development team to the components of the website.

13. A website development system according to claim 5, where the site is developed within the context of the website not as a separate entity.

14. A website development system according to claim 5, where the system restricts access to objects within a page in order to divide the development process.

15. A website development system according to claim 5, where a series of templates is constructed that present to the user only the information required to perform their assigned function within the system.

16. A website development system according to claim 5, where the templates are constructed in HTML with minimal extensions.

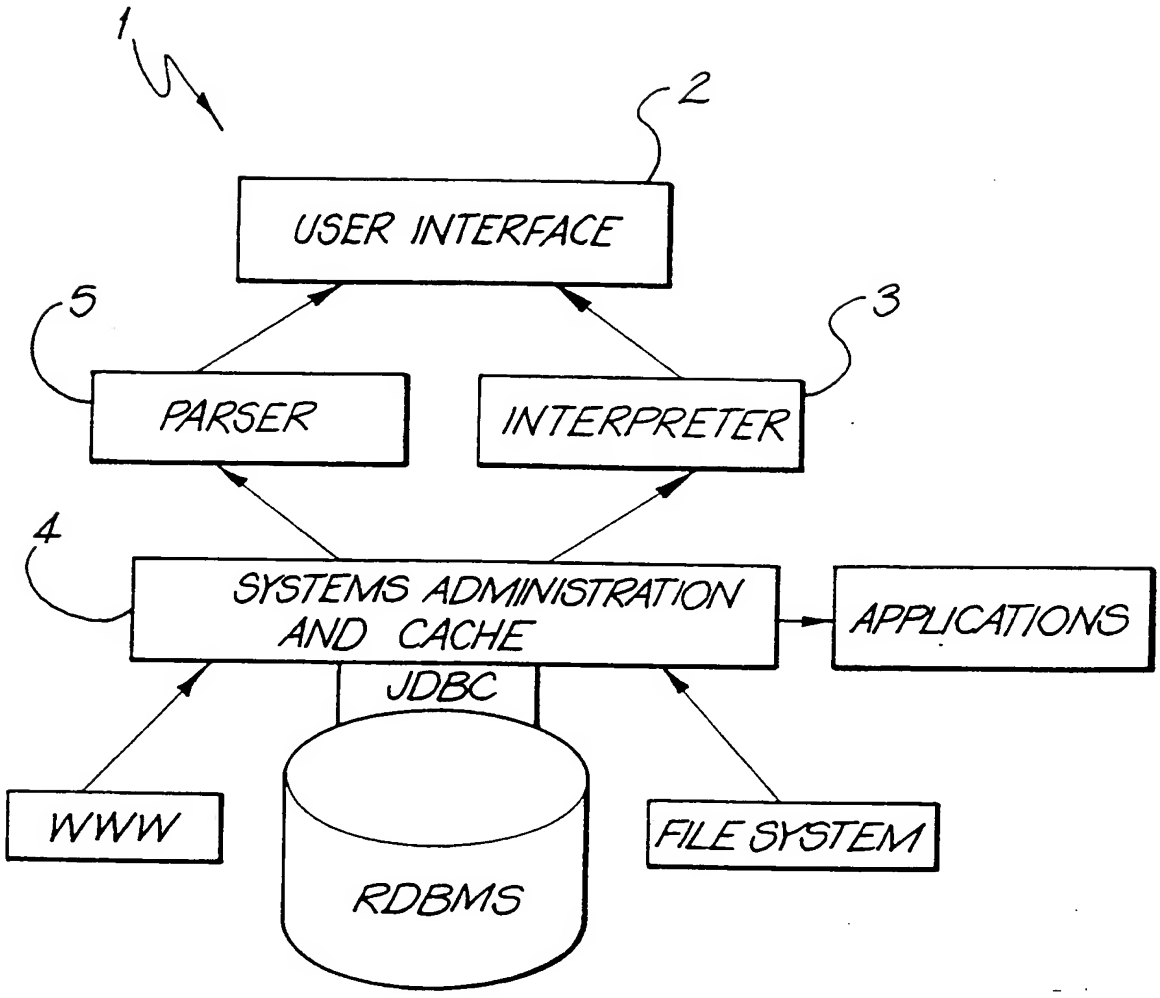


FIG. 1

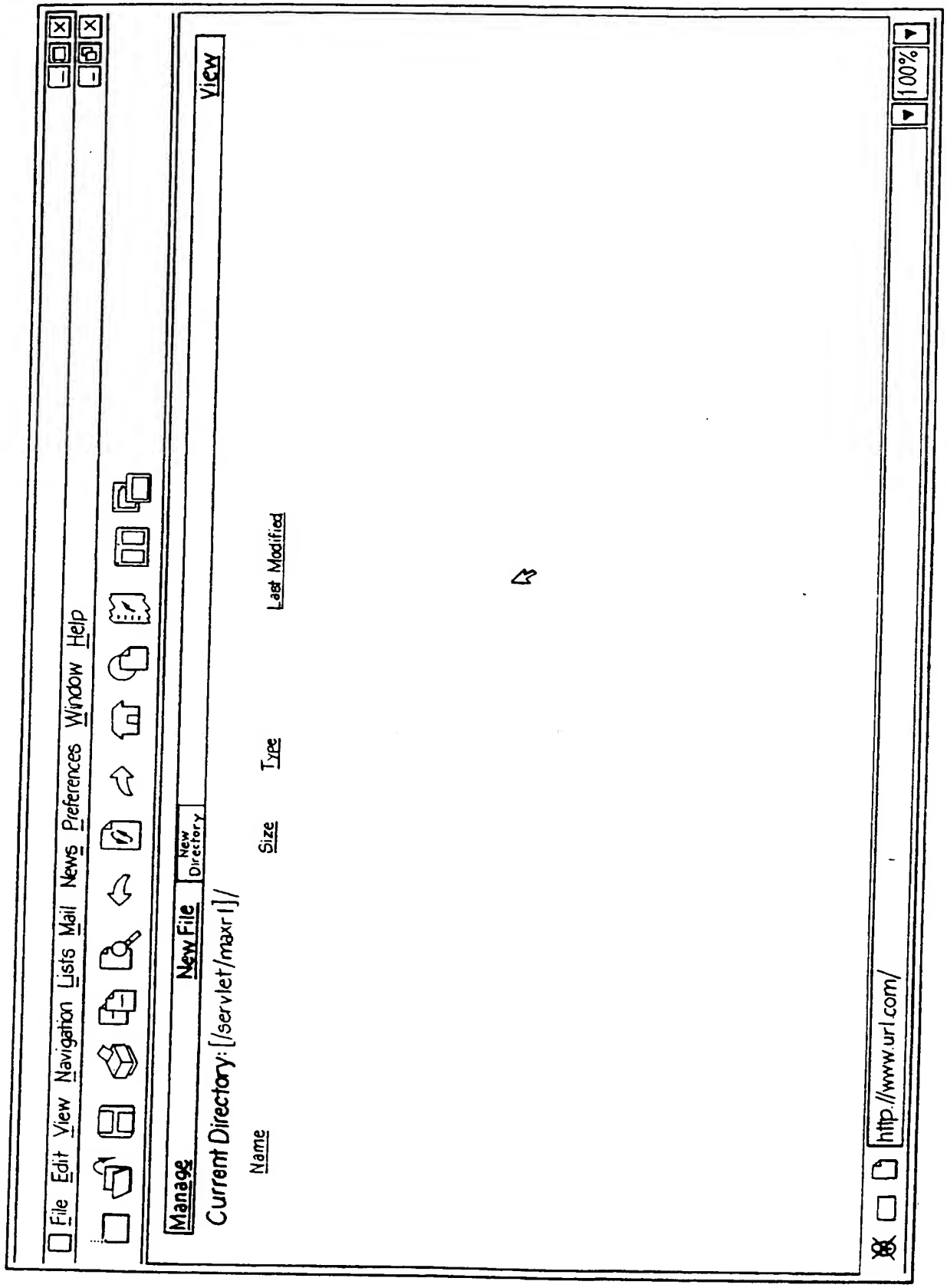


FIG. 2

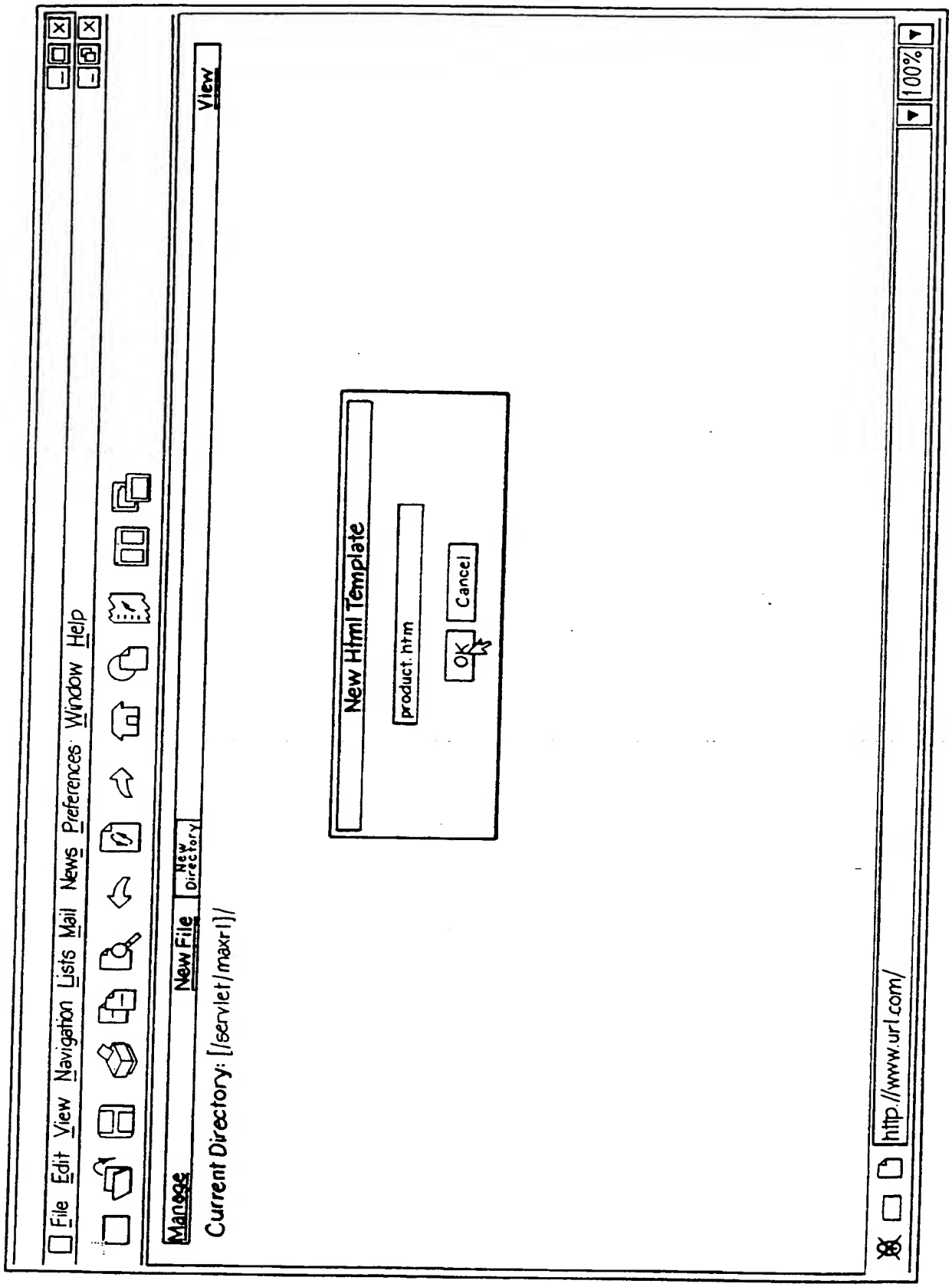


FIG. 3

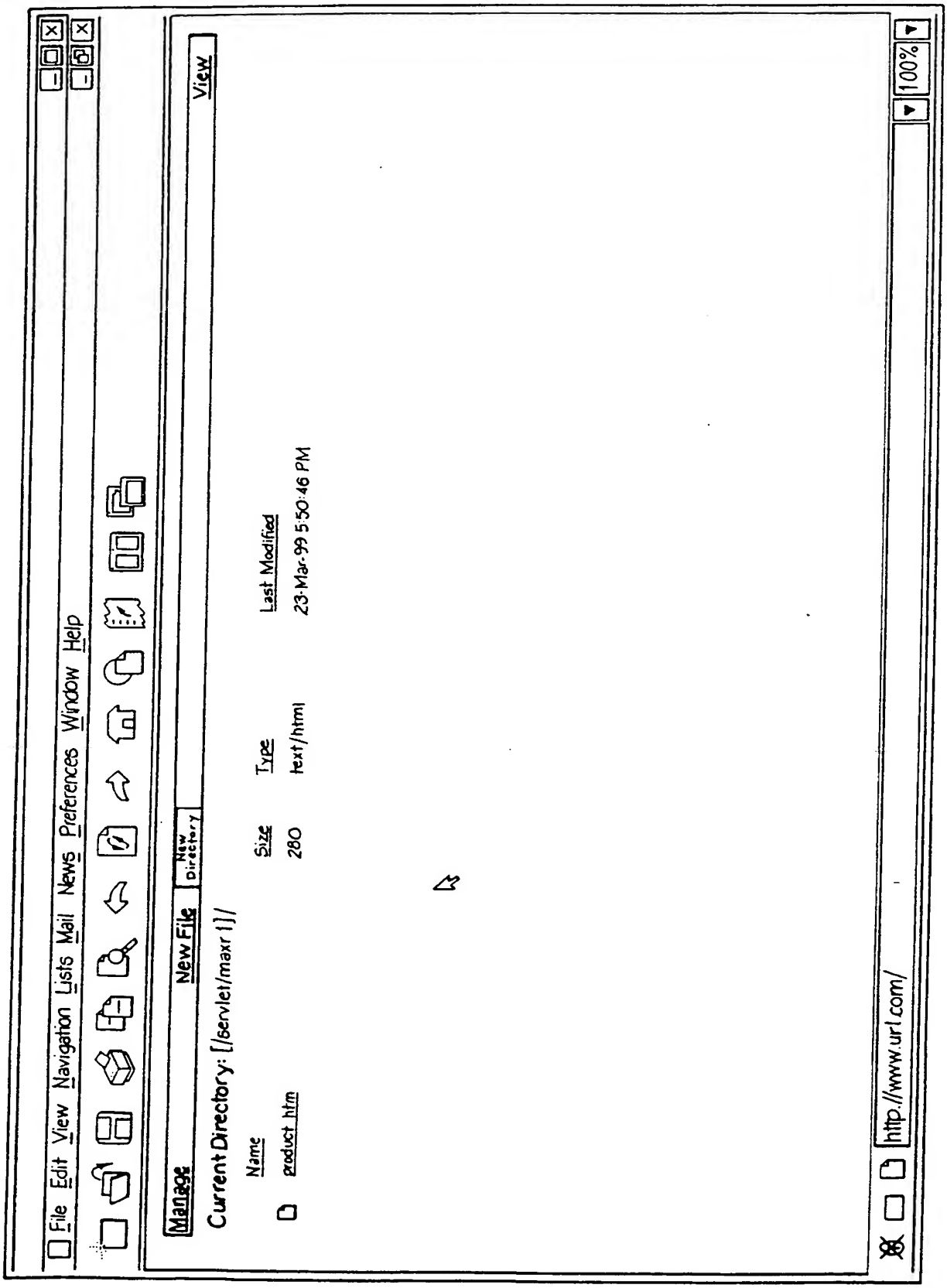


FIG. 4

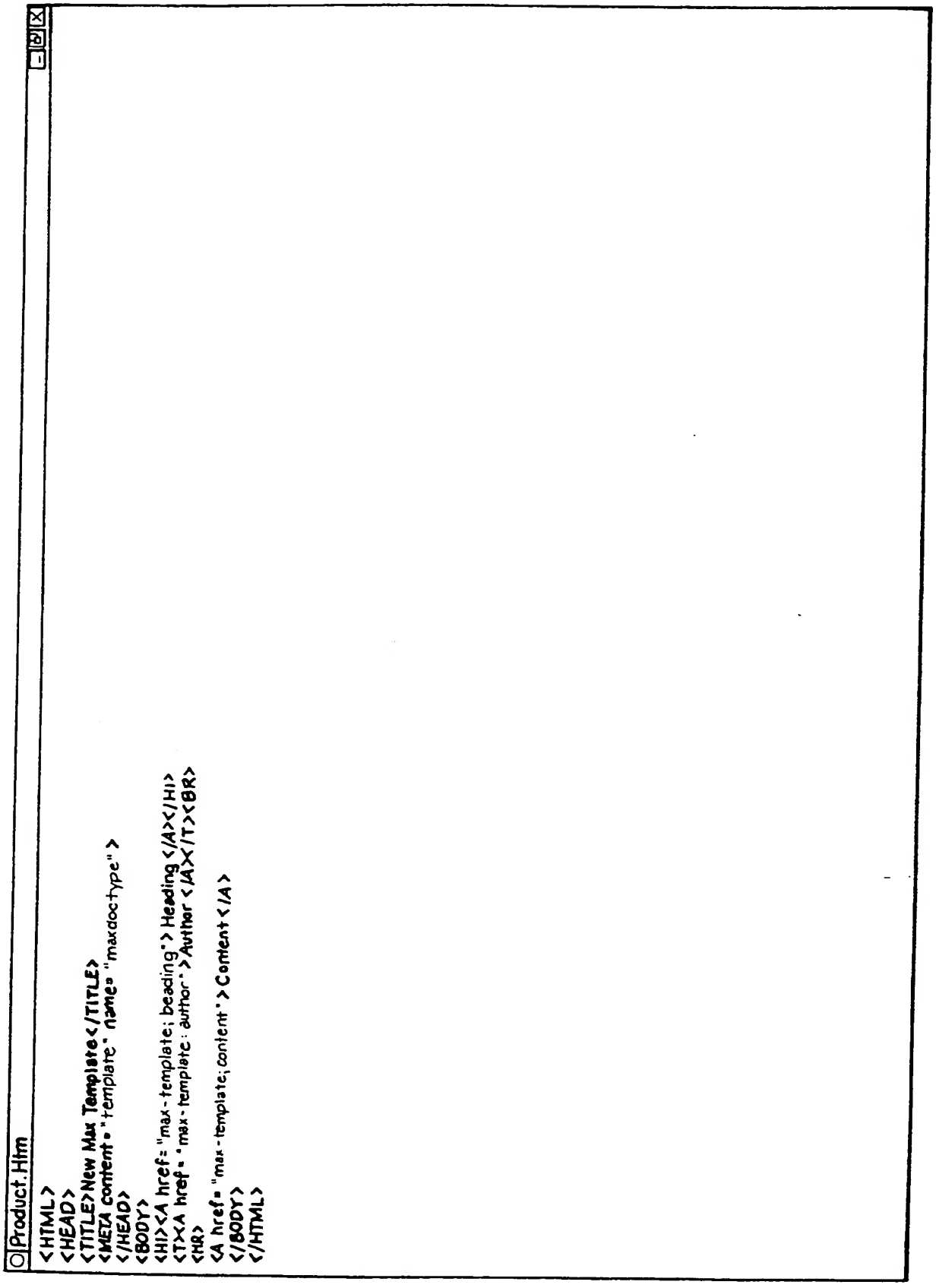


FIG. 5

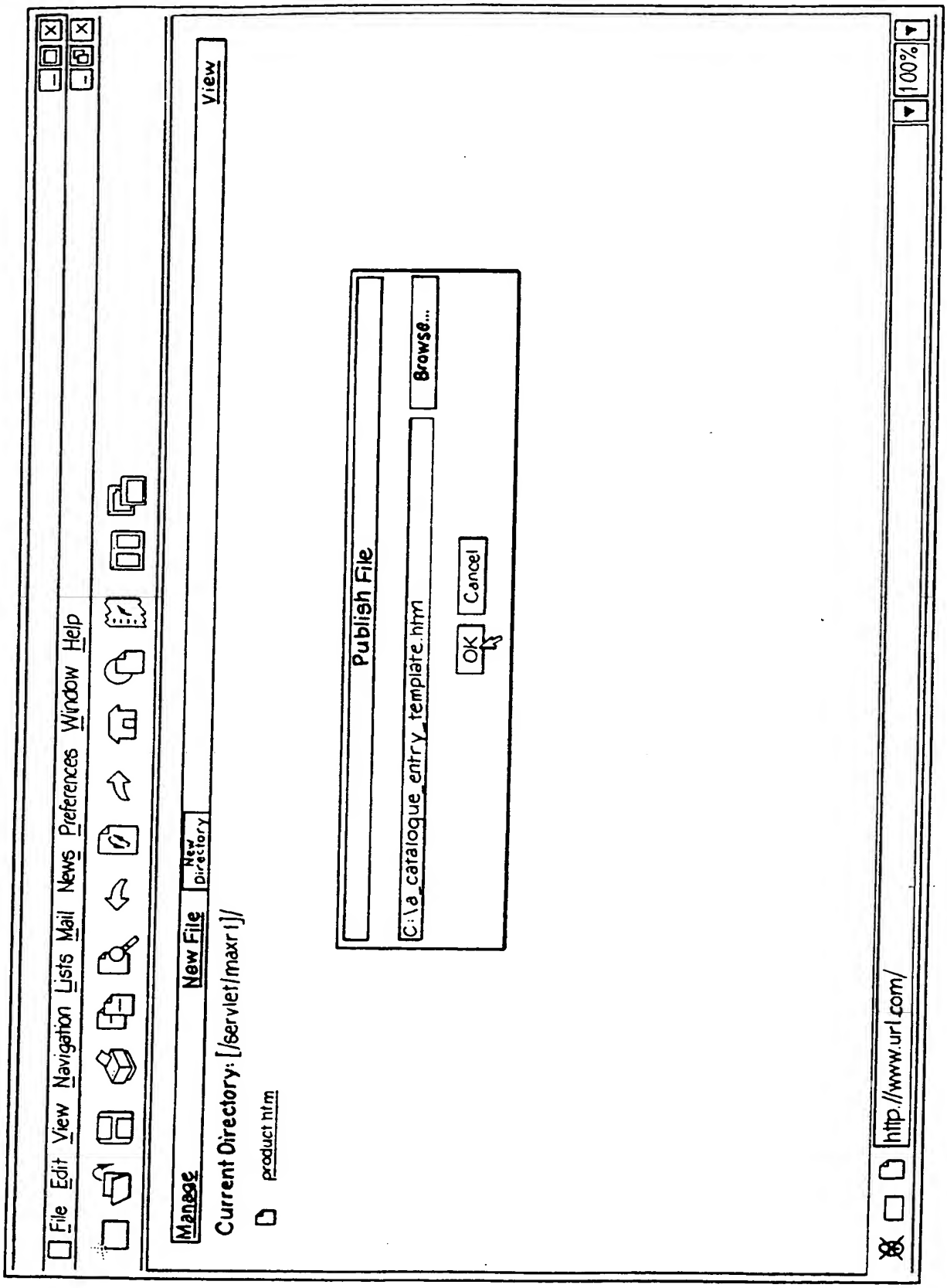


FIG. 6

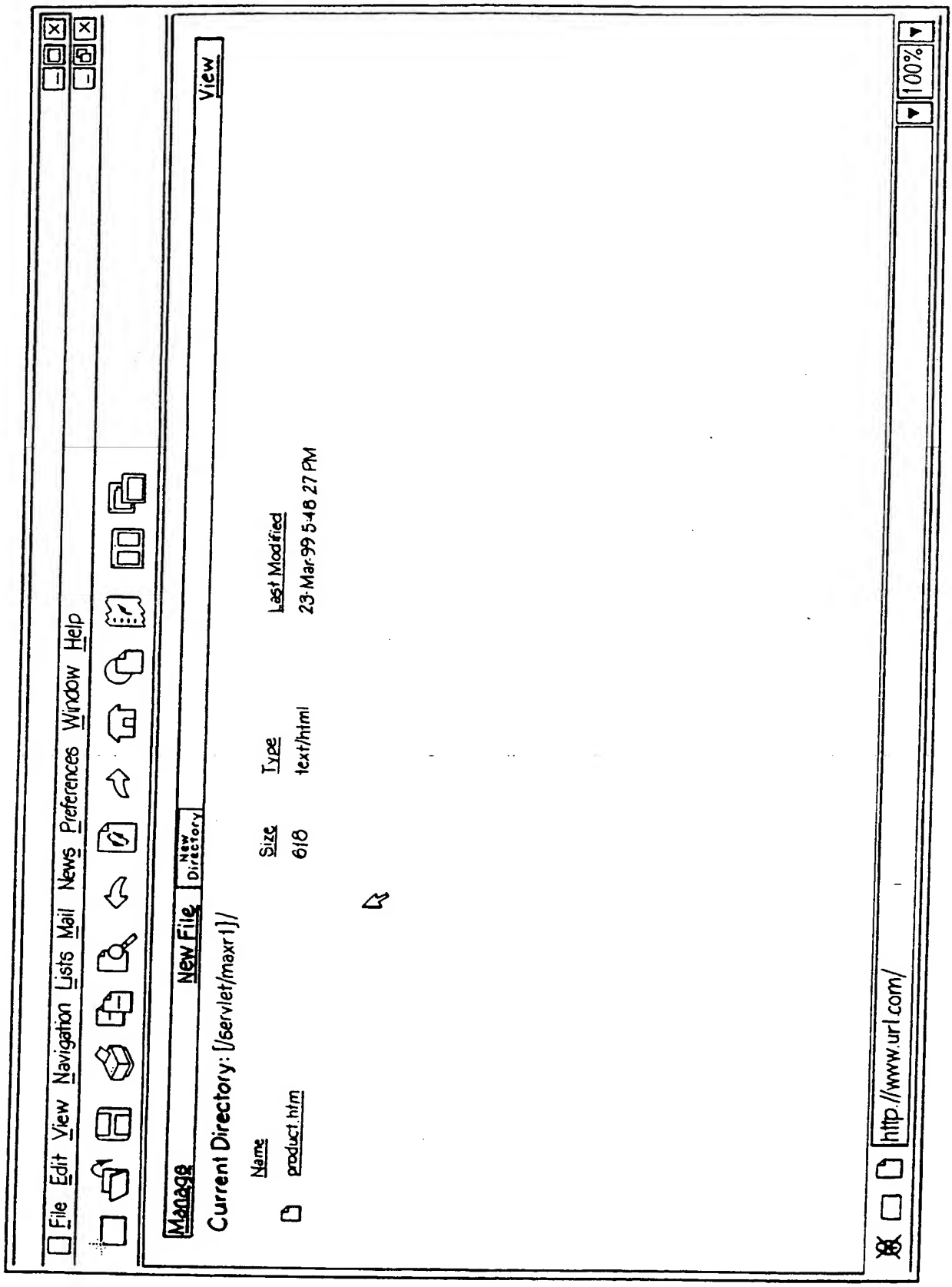


FIG. 7

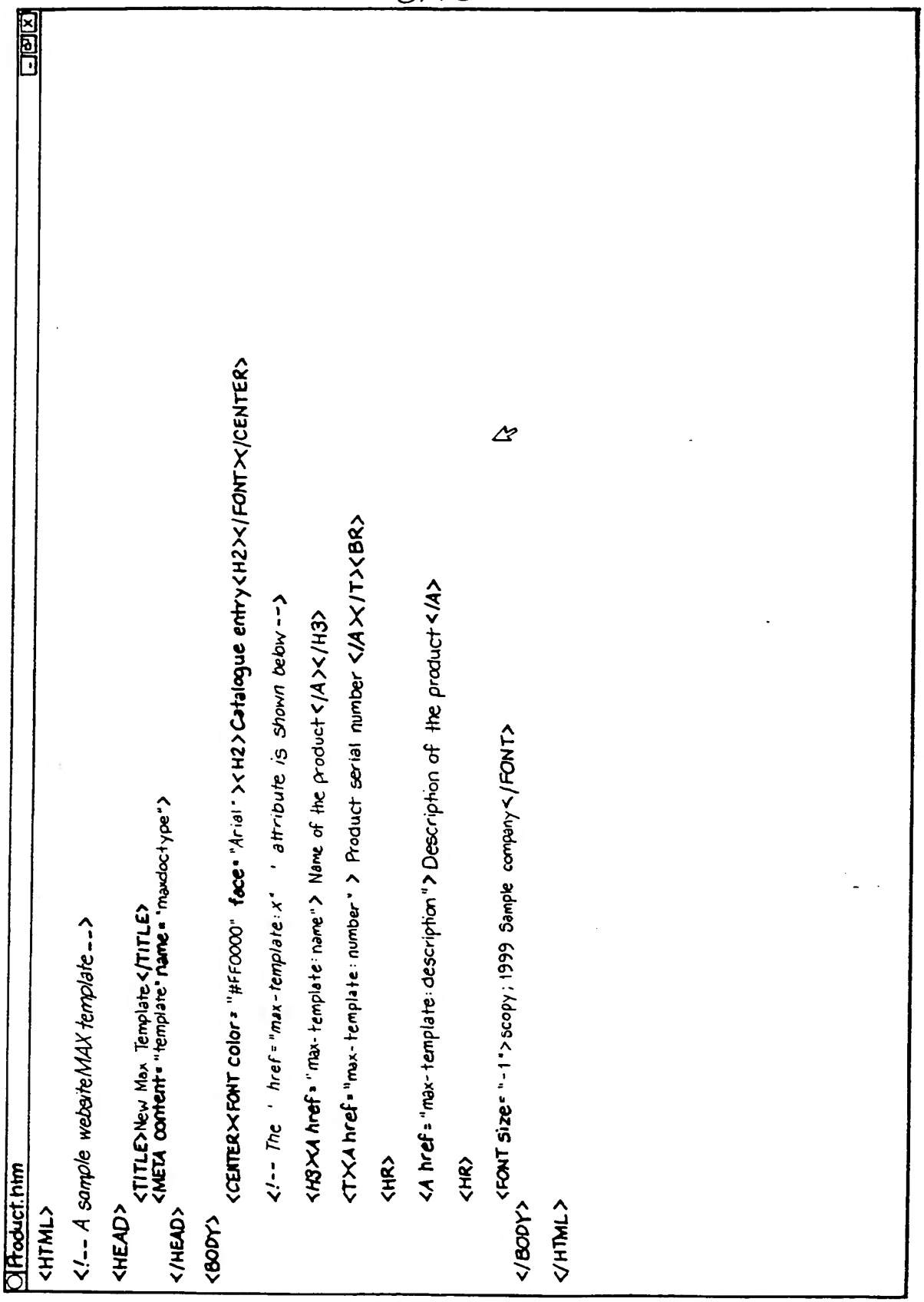


FIG. 8

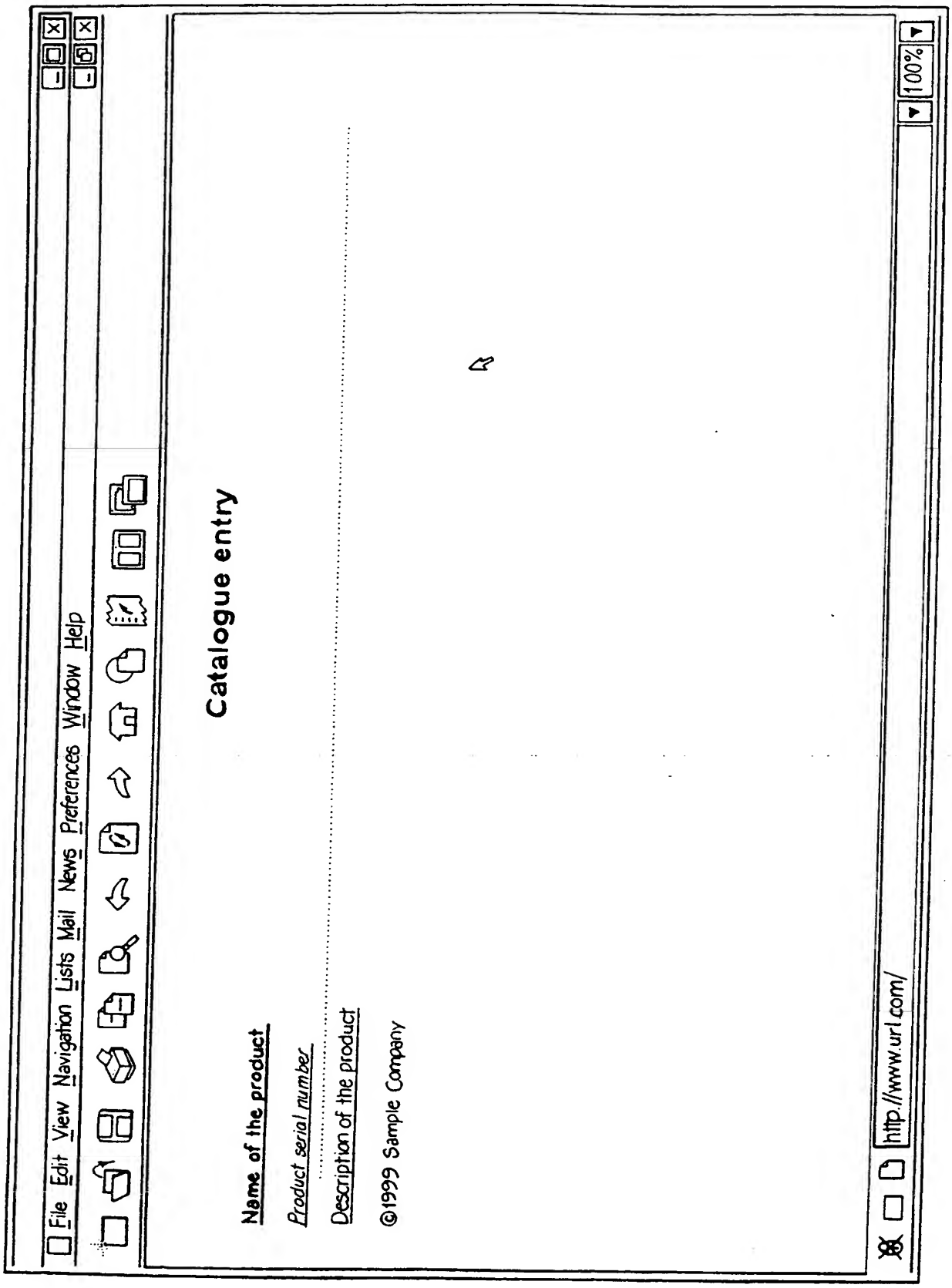


FIG. 9

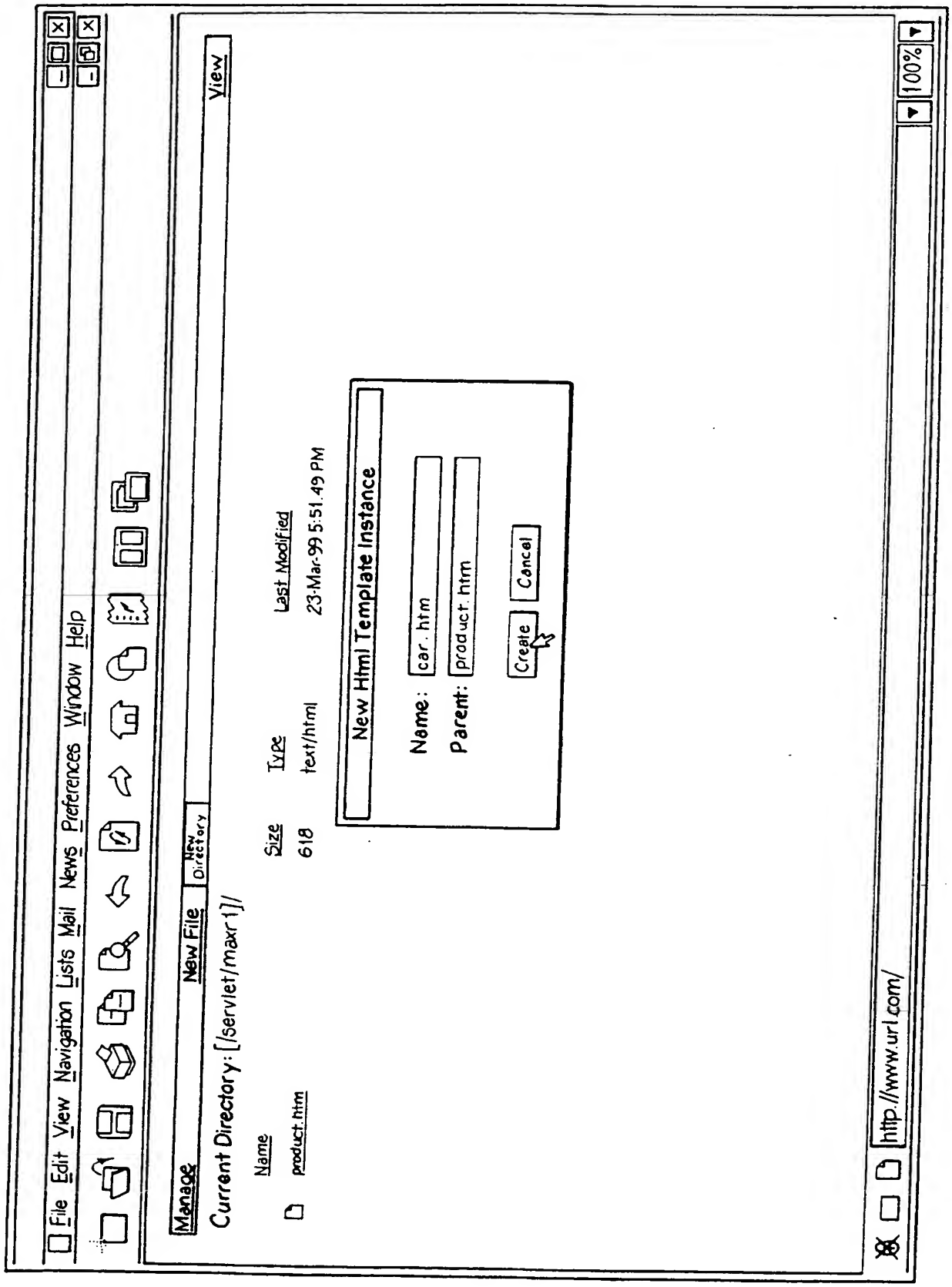


FIG. 10

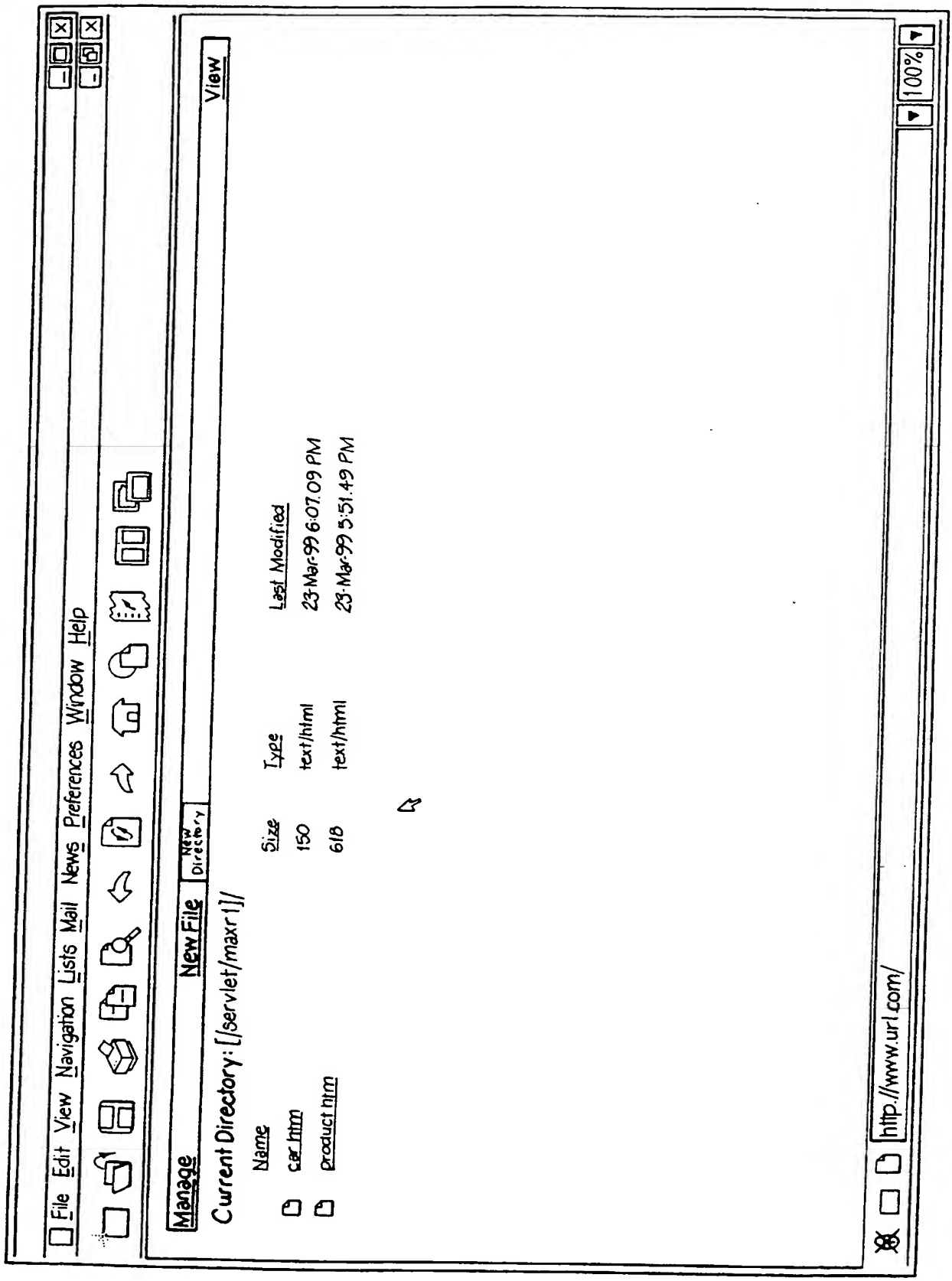


FIG. 11

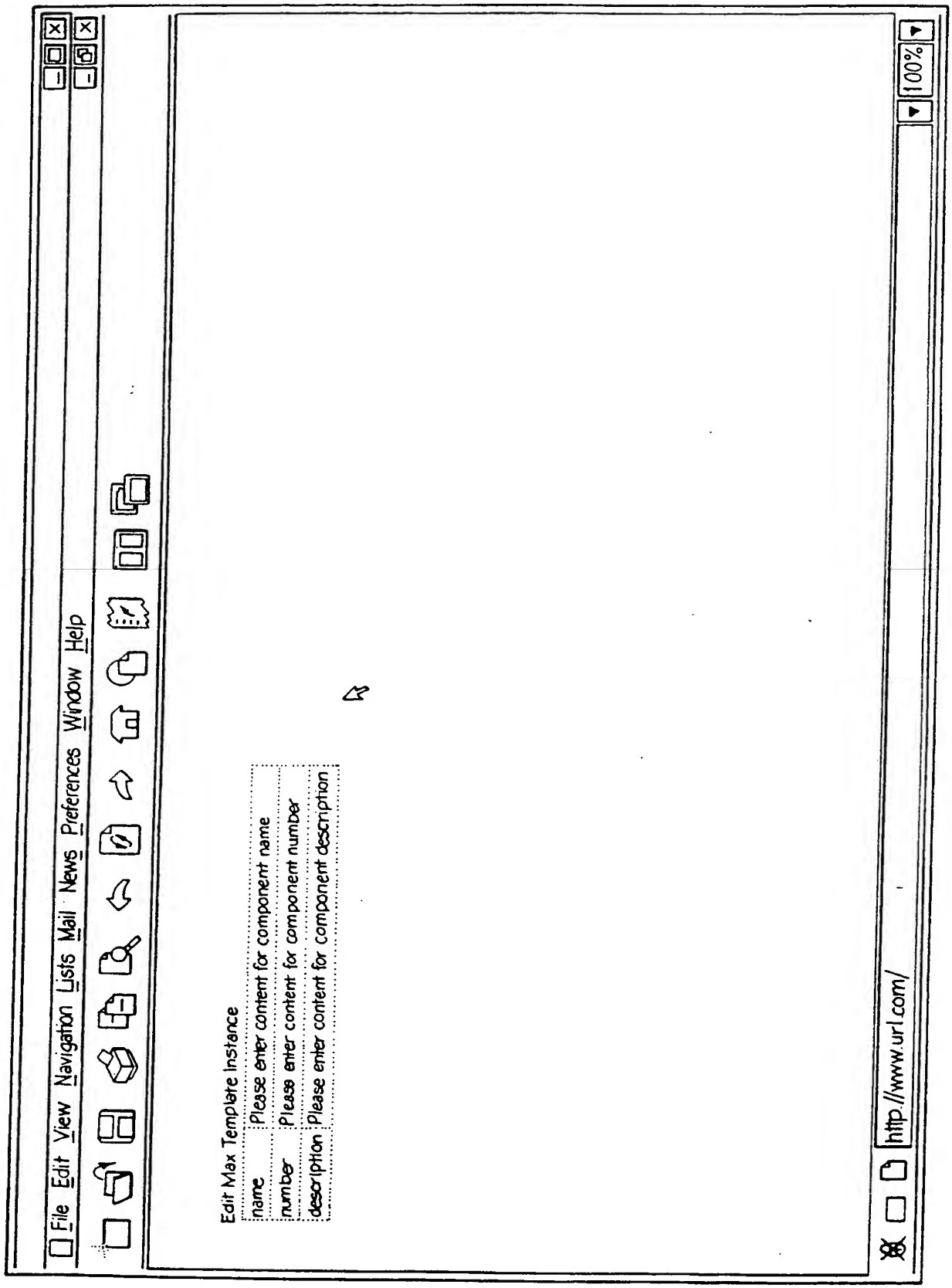


FIG. 12

13/16

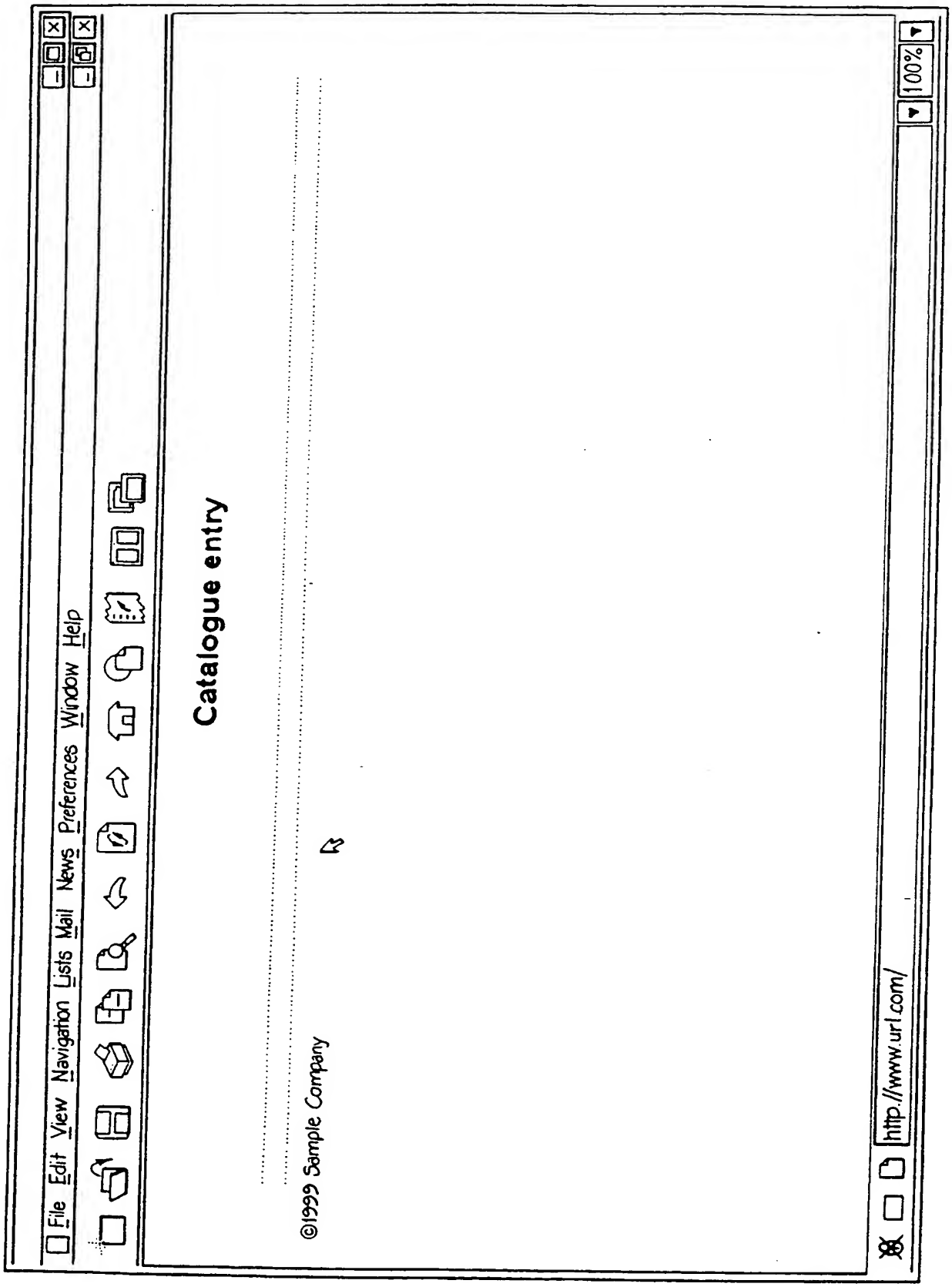


FIG. 13

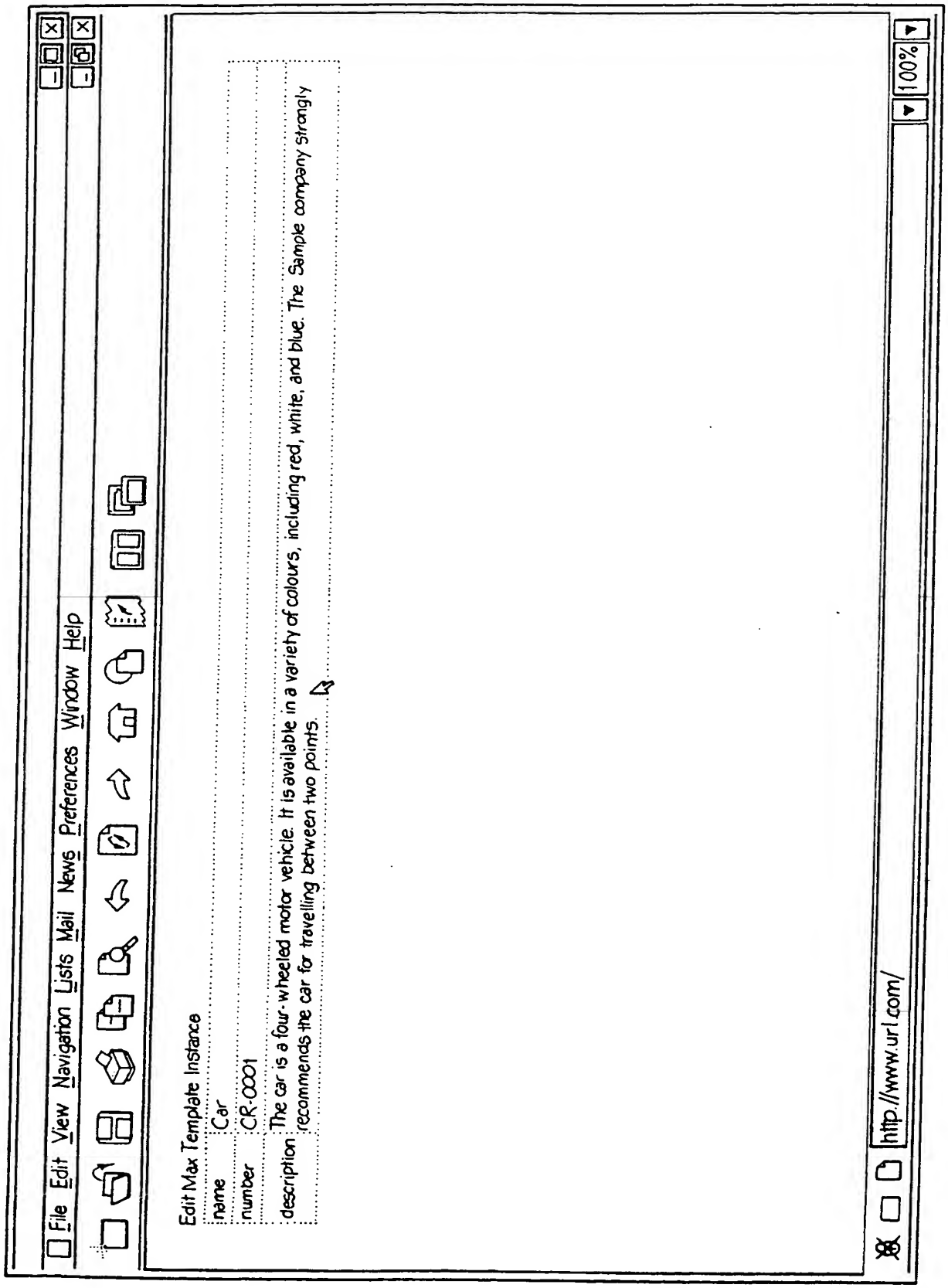


FIG. 14

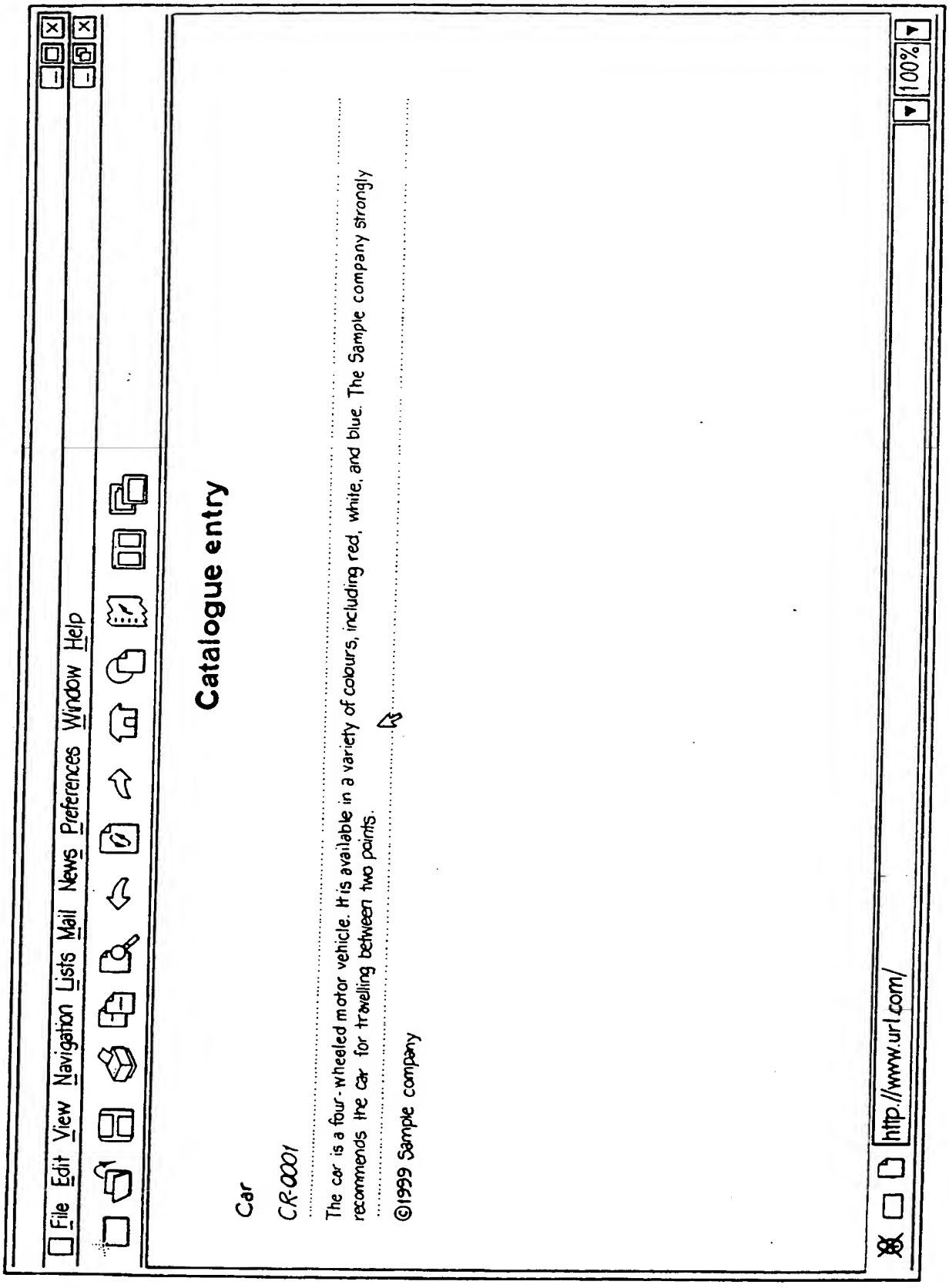


FIG. 15

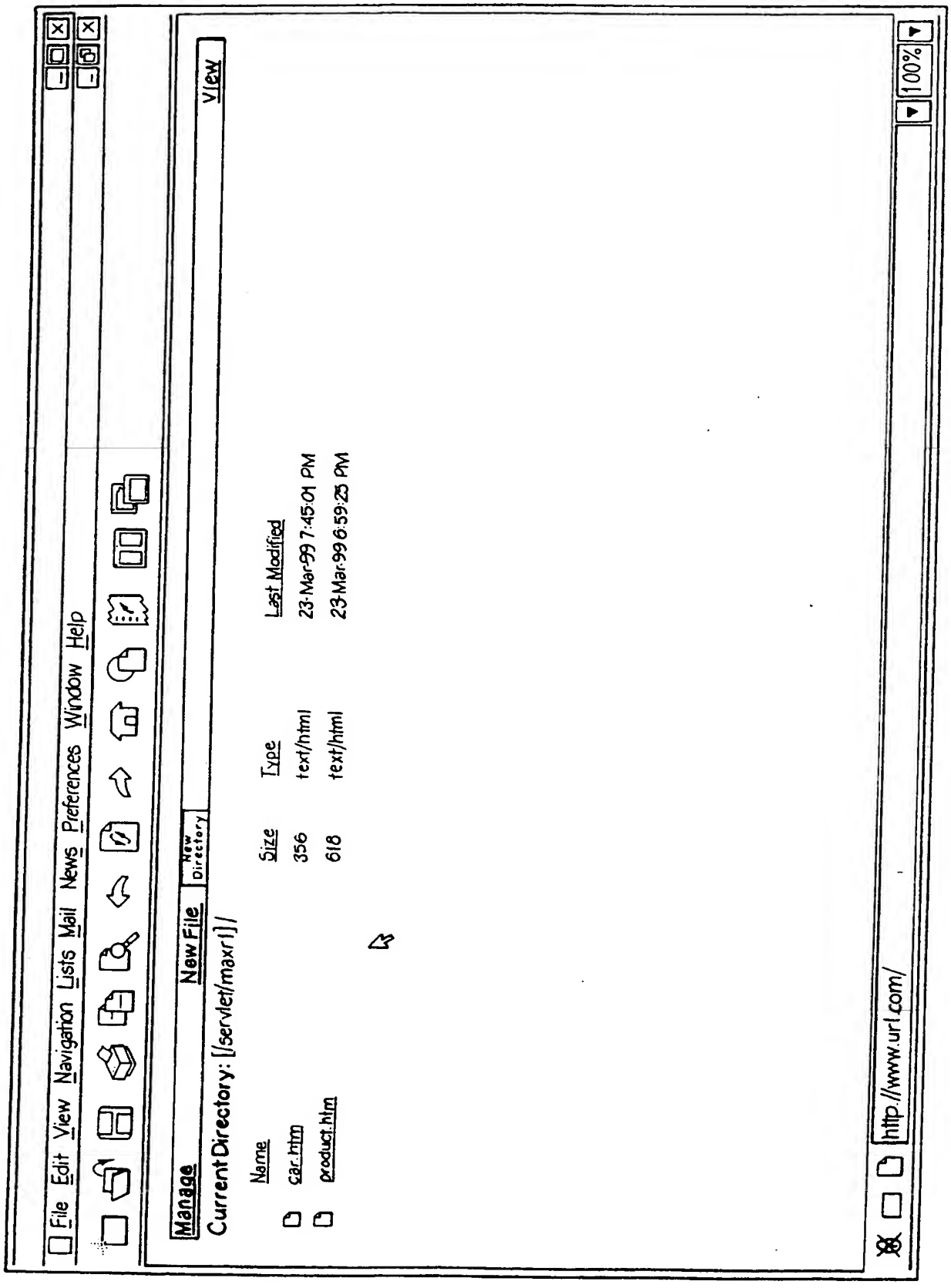


FIG. 16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU00/00468

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. [7]: G06F 17/60

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: G06F 17/60

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WPAT with keywords

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	WO 99/57657 A (LEXTRON SYSTEMS, INC) 11 November 1999	1 - 16
P,X	US 6026433 A (D'ARLACH et al) 15 February 2000	1 - 16
P,X	US 6035330 A (ASTIZ et al) 7 March 2000	1 - 16

☐ Further documents are listed in the continuation of Box C ☐ See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 August 2000	28 AUG 2000
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer **J.W. THOMSON** Telephone No : (02) 6283 2214